LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-PROC-609976

Recent Advances in the Mercury Monte Carlo Particle Transport Code

P. S. Brantley, S. A. Dawson, M. S. McKinley, M. J. O'Brien, D. E. Stevens, B. R. Beck, E. D. Jurgenson, C. A. Ebbers, J. M. Hall

January 14, 2013

International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)
Sun Valley, ID, United States
May 5, 2013 through May 9, 2013

Disclaimer

International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)
Sun Valley, Idaho, USA, May 5-9, 2013, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2013)

RECENT ADVANCES IN THE MERCURY MONTE CARLO PARTICLE TRANSPORT CODE

Patrick S. Brantley, Shawn A. Dawson, Michael Scott McKinley, Matthew J. O'Brien, David E. Stevens, Bret R. Beck, Eric D. Jurgenson, Chris A. Ebbers, James M. Hall
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, CA 94551
brantley1@llnl.gov; dawson6@llnl.gov; quath@llnl.gov, obrien20@llnl.gov; stevens9@llnl.gov; beck6@llnl.gov; jurgenson2@llnl.gov; ebbers1@llnl.gov; hall9@llnl.gov

ABSTRACT

We review recent physics and computational science advances in the *Mercury* Monte Carlo particle transport code under development at Lawrence Livermore National Laboratory. We describe recent efforts to enable a nuclear resonance fluorescence capability in the *Mercury* photon transport. We also describe recent work to implement a probability of extinction capability into *Mercury*. We review the results of current parallel scaling and threading efforts that enable the code to run on millions of MPI processes.

Key Words: Monte Carlo, Mercury, nuclear resonance fluorescence, parallel scaling, threading

1. INTRODUCTION

Mercury is a Monte Carlo particle transport code under development at Lawrence Livermore National Laboratory (LLNL) [1,2]. *Mercury* can transport neutrons, photons, and light element (hydrogen and helium) charged particles. Both fixed source and criticality problems are treated. Monte Carlo particles can be tracked through both combinatorial geometry and meshes. *Mercury* runs efficiently on current generation massively parallel computing platforms and is being improved to enable its use on emerging platforms. Previous conference papers have provided overviews of the capabilities and development of the *Mercury* code [3–7]. In this paper, we review recent physics and computational science advances in the *Mercury* code since the most recent update [3].

A major focus of physics development in *Mercury* has been to begin enabling the modeling of applications that require the treatment of nuclear resonance fluorescence. We describe the nuclear resonance fluorescence physics process and initial capabilities that have been developed to support this capability. We also describe the recent implementation of a probability of initiation capability in *Mercury* based on the Monte Carlo estimate of the probability of extinction. We present numerical results for a probability of initiation verification test problem.

A major focus of computer science development has been to enable *Mercury* to run on large, of order millions, of processes using both MPI and thread parallelism. We describe the current parallel scaling and threading efforts and show the results of initial scaling studies up to two million MPI processes.

2. NUCLEAR RESONANCE FLUORESCENCE

An effort is underway at LLNL to develop gamma-ray beam sources whose energies are tunable and nearly mono-energetic. These "MEGa-rays" (Mono-Energetic Gamma-rays) are obtained by Compton upscattering laser photons from high-energy electrons. The gamma-ray energy can be tuned to the known nuclear resonance fluorescence (NRF) energies of a desired isotope and used to interrogate samples for the presence of the isotope. Nuclear resonance fluorescence is a physical process by which a photon is absorbed by a nucleus which subsequently decays to its ground state through emission of one or more gamma rays with energies characteristic of the given isotope. The practical application of this technology requires the design of suitable photon detectors such as the Dual Isotope Notch Observer (DINO) detector [8]. Current design efforts at LLNL have used the COG Monte Carlo particle transport code [9]. The physics capabilities in *Mercury* are being improved to enable its use in ongoing detector design efforts. We note that the MCNPX code also has NRF capability [10].

The capability to use nuclear resonance fluorescence reaction cross sections through the Monte Carlo All-Particle Method (MCAPM) library [11] is currently being enabled in *Mercury*. The NRF reaction is a new reaction in addition to the Rayleigh (coherent) scattering, Compton (incoherent) scattering, photoelectric absorption, and pair production reactions already treated in MCAPM and *Mercury*. Details of the generation of ^{238}U NRF cross sections for use in the MCAPM library are given in the Appendix.

As an initial test of the *Mercury* NRF capability, we compared the modeling of photon reactions in *Mercury* and COG for a ^{238}U microdot test problem shown in Fig. 1. The microdot is a cylinder of diameter and length 1 μm (10^{-4} cm) composed of ^{238}U at a density of 10^{-3} g/cm^3. A mono-energetic incident beam of 680.1 keV photons impinges on the cylinder. Rayleigh, Compton, photoelectric, and NRF reactions are modeled (the source photons are below the threshold energy for pair production). The goal of the test problem is to tally the energy spectrum of the photons backscattered from the microdot. This problem has been modeled using both the *Mercury* and COG Monte Carlo transport codes with 10^8 Monte Carlo particles.

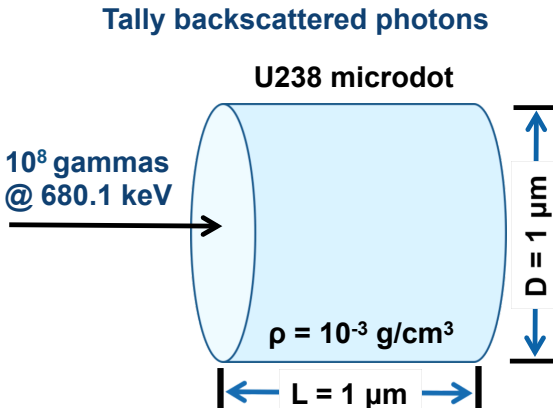

Figure 1. ^{238}U Microdot Problem

The energy spectra of the backscattered photons as computed by *Mercury* and COG are shown in Fig. 2, with relevant spectral features identified. The *Mercury* and COG backscattered spectra, in particular the

NRF emission lines, are in generally excellent agreement. The only significant discrepancy between the *Mercury* and COG results occurs for photons backscattered as a result of Rayleigh scattering (maintaining the source energy). This difference in backscattering appears to be related to a more accurate form factor treatment in COG and is currently under additional investigation. This microdot modeling comparison has resulted in improved modeling capability in both *Mercury* and COG. NRF reaction data for a broader range of isotopes is currently being generated.

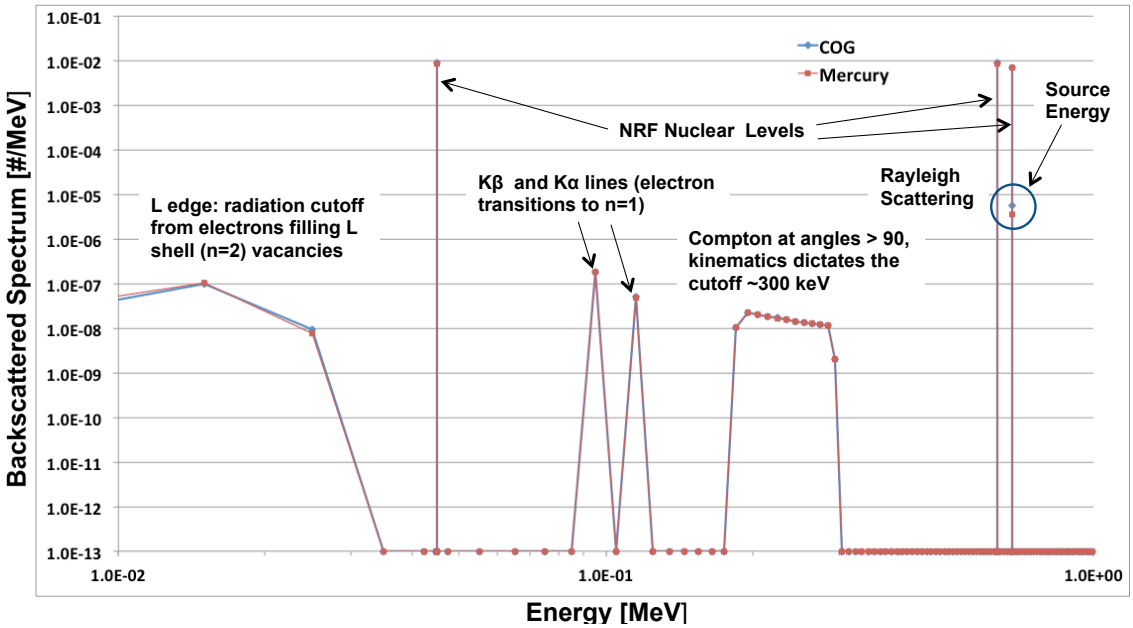

Figure 2. ^{238}U Microdot Backscattered Energy Spectrum

A photon linear polarization capability is also under development in *Mercury* to facilitate MEGa-ray detector design efforts. Details of this work will be reported in future publications.

3. PROBABILITY OF EXTINCTION

In a supercritical nuclear system, a single neutron can result in a divergent chain reaction. Alternatively, a single neutron can lead to a finite length neutron chain that goes extinct as a result of neutron absorption and leakage. The probability of initiation (*poi*) is the probability that one neutron from a specified source leads to a divergent chain reaction, while the probability of extinction (*poe*) is the probability that it does not lead to a divergent chain reaction (*poe* = 1 - *poi*).

A *poi* capability based on a forward solution to the Boltzmann transport equation was previously developed and implemented in *Mercury* [12]. A *poi* simulation in *Mercury* begins with a source of particles that represent the head of unique families of neutrons. A family originates with a single source particle. Each time cycle, the number of progeny for each family is calculated. Once the number of progeny has surpassed a predefined user-supplied threshold, the family is tallied as divergent and the whole family is

removed. The probability of initiation is computed as the number of divergent families divided by the number of initial source families. Rising et al. [13] have also recently implemented time-based and generation-based *poi* cutoff methods in a research version of the MCNP code.

Recently, Booth [14] suggested the use of a *poe* estimate computed using Monte Carlo to compute the probability of initiation, and he implemented this *poe* method in a modified version of MCNP [15]. One of the main advantages of the *poe* method over the *poi* method is that it does not require a user to specify the number of progeny required to determine if a chain is divergent. In addition, Booth has demonstrated useful variance reduction approaches for the *poe* algorithm.

A *poe*-based capability for determining the probability of initiation has recently been implemented in *Mercury* [16]. The *poe* implementation in *Mercury* is time-based rather than generation-based. As in the *poi* case, a *poe* simulation in *Mercury* begins with a source of particles that represent the heads of unique families of neutrons. Each time cycle, the number of families that have gone extinct are counted. The probability of extinction is computed as the number of extinct families divided by the number of initial source families. The probability of initiation is then computed as *poi* = 1 - *poe*. The *poe* algorithm has also been extended to accommodate the spatial parallelism via domain decomposition that is available in *Mercury* [17].

As an initial verification of the *Mercury poe* capability, we compute the solution to an analytic infinite medium problem proposed by Booth [14]. The material has only two reactions, absorption and fission. The probability per collision for an absorption is 49.9%, while the probability for fission is 50.1%. Fissions always produce exactly two neutrons. The analytic *poi* for this test problem is 3.992016×10^{-3}.

The *Mercury* results for the *poi* and *poe* methods are shown in Table I, where N is the number of starting families and the *poi* computed using the *poe* algorithm is given by *poi* = 1 - *poe*. Both methods exhibit convergence towards the analytic solution with increasing number of starting families. (Results with over 2^{21} = 2,097,152 starting families are not available, because the *Mercury* simulations run out of computer memory.) Differences with Booth's results may be due to the time-based implementation in *Mercury* compared to the generation-based approach in Booth's implementation. These initial verification results support the correctness of both the *poi* and *poe* implementations in *Mercury*.

4. PARALLEL SCALING AND THREADING

4.1. Parallel Algorithm Scaling

A substantial amount of effort has recently been devoted to improving the scalability of *Mercury* to large (of order millions) numbers of parallel processes. Here we define *scalability* as the ability of a code to perform efficiently as the number of parallel processes increases. An example of a code that is *non-scalable* is one that exhibits a run time proportional to the number of processes. An example of a code that is *scalable* is one that exhibits a run time proportional to the logarithm of the number of processes. Enabling a code to be scalable to millions of processes requires attention to the details of algorithms and memory usage that can be safely ignored when considering scaling to only a few thousand processes.

The specific *Mercury* algorithms currently being improved for scalability include the algorithms for sourcing particles, for globally resolving particle locations onto the correct process (for Cartesian domains), for load balancing the particle workload across processes (for the replication-only case [7]), and

Table I. Analytic POI Test Problem Results

N	*poi* ($\times 10^{-3}$)	Fractional Error	*poe*'s *poi* ($\times 10^{-3}$)	Fractional Error
2^{10}	3.906	-0.021	2.138	-0.464
2^{11}	5.371	0.345	4.147	0.039
2^{12}	3.906	-0.021	2.893	-0.275
2^{13}	3.662	-0.083	3.710	-0.071
2^{14}	5.005	0.254	4.782	0.198
2^{15}	4.456	0.116	4.640	0.162
2^{16}	4.211	0.055	4.478	0.122
2^{17}	4.120	0.032	4.268	0.069
2^{18}	3.944	-0.012	4.056	0.016
2^{19}	3.937	-0.014	4.046	0.014
2^{20}	3.966	-0.006	4.029	0.009
2^{21}	3.965	-0.007	3.977	-0.004
∞	3.992	–	3.992	–

for deciding at what point particle streaming communication has completed. In addition to addressing parallel scalability of the algorithms themselves, the memory usage of the various algorithms has also been carefully scrutinized to enable scalability. Table II summarizes the previous and improved parallel scalability for various algorithms in *Mercury*, where N_{proc} is the number of processes.

Table II. Parallel Scalability of Monte Carlo Algorithms for N_{proc} Processes

Algorithm	Previous Scaling	Improved Scaling
Particle Sourcing	$O(N_{proc})$	O(1)
Global Particle Find - Cartesian Domains	$O(N_{proc})$	$O((\log N_{proc})(\log\log N_{proc}))$
Load Balancing - Replication	$O(N_{proc}^2)$	$O(\log N_{proc})$
Test for Done with Particle Communication	$O(N_{proc}^2)$	$O(\log N_{proc})$

The previous particle sourcing algorithm looped on each process over the total global number of Monte Carlo source particles and sourced the particles onto the current process if the particle belonged on the domain associated with the process. Assuming the number of source particles scales linearly with the number of processes N_{proc} (i.e. weak scaling), this particle sourcing algorithm scales as $O(N_{proc})$. The improved scalable algorithm loops over the local number of particles (total number of source particles

divided by the number of processes) and sources all local particles but requires subsequent communication to locate the particles on the correct process. We have developed a new scalable algorithm to locate the particle on the correct process that scales as O((log N_{proc})(log log N_{proc})). Details of this improved algorithm will be published in a future paper.

The previous load balancing algorithm [18] gathered a global view of the amount of work on each process and then made global decisions about how to communicate particles to achieve load balance. That load balancing algorithm scales as O(N_{proc}^2) and is not scalable, since it requires information about every process. The improved load balancing algorithm [19] performs an iterative pair-wise (process pairs) balancing step such that the particle workload is balanced across all processes after O(log(N_{proc})) iterations.

Finally, we have implemented in *Mercury* a scalable "test-for-done" algorithm [20] to determine when all particle communication has been completed. The previous algorithm stored an $N_{proc} \times N_{proc}$ matrix of the number of sent and received messages, making the algorithm O(N_{proc}^2) and hence not scalable. The scalable algorithm uses tree-based communication and non-blocking reduce and broadcast operations concurrently as particles are being tracked, resulting in O(log(N_{proc})) scaling.

As an initial demonstration of the scalability of the *Mercury* parallel implementations, we performed a weak scaling study using the Godiva critical sphere benchmark problem (HEU-MET-FAST-001) [21]. This problem consists of a sphere of radius 8.7407 cm composed of highly-enriched uranium at a density of 18.740 g/cm^3 and with isotopic atom fractions: ^{234}U = 0.01025002, ^{235}U = 0.9376829, and ^{238}U = 0.05206708. A weak scaling study was performed ranging from 2^6 = 64 to 2^{21} = 2,097,152 MPI processes on the Sequoia supercomputer at LLNL, a 20-petaFLOP/s IBM computer with 16 PPC A2 CPU cores per compute node, four hardware threads per core, and 16 GB memory per node. The weak scaling study was performed using pure MPI parallelism and 10^4 Monte Carlo particles per process, resulting in up to approximately 21 billion Monte Carlo particles being tracked for the 2^{21} process case.

The wallclock time spent tracking particles as a function of the number of processes is shown in Table III. The tracking wallclock time is remarkably constant, varying by less than 7%, as the number of processes ranges from 64 up to 2,097,152.

The results of this initial weak scaling study demonstrate the scalability of *Mercury* to very large numbers of MPI processes. Additional studies using problems with large numbers of cells, nuclides, and/or tallies may reveal additional algorithms that require improvements to scale in these orthogonal code dimensions. But we believe that the *Mercury* parallel process scaling is largely independent of these other code scaling dimensions.

4.2. Threading

Mercury has traditionally used pure MPI to achieve parallelism on massively parallel computers. MPI is used across compute nodes of the computational platform as well as on the CPU cores of individual nodes. The ability to use threads with the OpenMP standard has recently been added to *Mercury* to enable parallelism across the cores of each node. Simulations can now use a combination of both MPI and OpenMP to distribute the Monte Carlo work across the CPU cores of a node. Each node may have one or more MPI processes, and each MPI process may use one or more threads (tasks) to access compute cores on the node. The threading capability can potentially result in significant memory savings, because nuclear

Table III. Tracking Wallclock Time [s] for Godiva Critical Sphere Weak Scaling

Number Processes N_{proc}	$\log_2(N_{proc})$	Tracking Wallclock Time
		[s]
64	6	257
128	7	258
256	8	261
512	9	261
1,024	10	263
2,048	11	263
4,096	12	264
8,192	13	264
16,384	14	264
32,768	15	267
65,536	16	267
262,144	18	270
524,288	19	274
1,048,576	20	272
2,097,152	21	274

data can be stored once per MPI process instead of once per CPU core. This memory savings is expected to be important as future computer platforms move toward a larger numbers of nodes and processor cores per node coupled with lower memory available per node.

Mercury has three methods of distributing the work across the CPU cores of a computer: 1) spatial decomposition [7,17], 2) spatial replication [7,18], and 3) now OpenMP work sharing. The first two approaches are used to distribute work across MPI processes. The third approach is used to distribute the work of a single MPI process across multiple CPU cores within a single compute node. OpenMP pragmas are used to thread over particles, cells, and other sections of the code that perform significant work. OpenMP may be used with both spatial decomposition and spatial replication. We note that MCNP has a similar combined MPI/OpenMP parallelism for the replication-only case [22].

Coarse grain threading is achieved by creating a particle *vault* (list of particles to be tracked) for each thread and distributing particles evenly across the vaults. At a high level, each thread works on the particles in its vault. Implementing this capability requires a thread (task) layer in many data structures to enable multiple threads to operate on particles independently without requiring thread locks that can degrade efficiency. At the end of the particle transport, non-threaded code sums tallies over task layers to task zero. *Mercury* also uses fine grained OpenMP parallelism at lower loop levels outside of the particle processing loop.

To examine the efficiency of the *Mercury* threading implementation, we performed a scaling study using the Godiva critical sphere benchmark problem (HEU-MET-FAST-001) [21] described above. The

simulation was performed using continuous energy LLNL ENDL2009 nuclear data and 10^7 Monte Carlo particles per static-K generation. Fifteen generations were discarded prior to averaging the eigenvalue, and the eigenvalue fractional convergence tolerance was set to 2.5×10^{-4}. In all cases described below, *Mercury* computed k_{eff} = 1.000188±0.000166 which agrees with the experimental value of 1.000±0.001 [21] to within statistics. We performed a strong scaling study in which we fixed the problem size (geometry and number of Monte Carlo particles) and varied the number of processes used to simulate the problem from 16 to 1,024. The simulations were performed on the LLNL RZMerl Linux cluster that has 16 Intel Xeon (Sandy Bridge) cores (2.6 GHz) and 32 GB of memory per core. We investigated the impact on the particle tracking time and maximum node memory of the number of threads per node used (varied from two to sixteen) for each of the total number of CPU values.

The wallclock time [s] spent tracking particles is shown in Table IV. The strong scaling of *Mercury* for this problem is excellent - doubling the number of processors consistently produces a factor of two decrease in tracking time. For a given number of CPUs, the particle tracking time is generally insensitive to the number of threads used except when using sixteen threads. These results demonstrate that the threading implementation for particle tracking is essentially as efficient as using MPI processes within a node. A clear degradation in performance does occur when using the same number (sixteen) of threads as total CPUs on the node. The wallclock times shown in Table IV exclude the initialization and finalization times; including those times introduces up to approximately a 20% loss of efficiency in the threading results with up to eight threads per node. This degradation in efficiency implies that additional opportunities for threading potentially exist in the initialization and finalization portions of the code.

Table IV. Tracking* Wallclock Time [s] for Godiva Critical Sphere Problem

	Number MPI Processes / Number Threads Per MPI Process				
Number CPUs	16 / 1	8 / 2	4 / 4	2 / 8	1 / 16
16	250.8	247.7	252.3	252.1	265.9
32	124.0	125.8	125.2	125.8	134.5
64	62.3	62.8	63.0	63.0	66.6
128	31.2	31.8	31.6	31.5	33.2
256	15.6	15.7	15.7	15.7	16.8
512	7.8	7.9	7.9	7.9	8.4
1,024	3.9	3.9	3.9	3.9	4.2

* Excludes initialization and finalization

The maximum node memory [GB] at the end of the calculation is shown in Table V. For this problem, the memory reduction per MPI process elimination (in favor of using a thread) appears to be approximately 90-100 MB. This memory reduction is largely attributable to storing fewer copies of the nuclear data in memory. Because this is a strong scaling problem with a fixed number of Monte Carlo particles, the maximum node memory decreases as the number of CPUs is increased as a result of the decrease in the number of particles per CPU.

Table V. Maximum Node Memory [GB] for Godiva Critical Sphere Problem

	Number MPI Processes / Number Threads Per MPI Process				
Number CPUs	16 / 1	8 / 2	4 / 4	2 / 8	1 / 16
16	3.22	2.47	2.11	1.93	1.84
32	2.59	1.81	1.43	1.20	1.18
64	2.16	1.35	0.95	0.72	0.67
128	1.95	1.12	0.71	0.48	0.34
256	1.85	1.00	0.59	0.37	0.24
512	1.80	0.95	0.53	0.31	0.18
1,024	1.79	0.93	0.50	0.28	0.15

From the results of this scaling study, we conclude that the use of MPI processes in conjunction with OpenMP threads in *Mercury* produces similar computational performance to using MPI alone but with a significant reduction in memory usage.

5. CONCLUSIONS

We have described recent physics and computer science advances in the *Mercury* Monte Carlo particle transport code. The addition of nuclear resonance fluorescence capabilities into *Mercury* is aimed at enabling the application of the code to ongoing detector design efforts. A probability of initiation capability based on a probability of extinction algorithm has been implemented in *Mercury* as an alternative to the probability of initiation algorithm. Significant improvements in the parallel scalability of various algorithms and the enabling of an OpenMP threading option position *Mercury* to run efficiently on the current state-of-the-art computer platforms with large numbers of processors and low memory per processor.

ACKNOWLEDGEMENTS

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. A part of this work was funded under the LLNL LDRD project 12-ERD-060.

REFERENCES

[1] P. S. Brantley, M.S. McKinley, “Mercury Web Site,” http://mercury.llnl.gov (2012).

[2] P. S. Brantley, S. A. Dawson, J. P. Grondalski, M. S. McKinley, M. J. O’Brien, R. J. Procassini, S. M. Sepke, D. E. Stevens, T. B. Yang, *Mercury User Guide: Version d.10*, LLNL-SM-560687 (Revision #2), Lawrence Livermore National Laboratory Report (2012).

[3] R. Procassini, P. Brantley, S. Dawson, G. Greenman, M. S. McKinley, M. O'Brien, S. Sepke, D. Stevens, B. Beck, C. Hagmann, "New Features of the Mercury Monte Carlo Particle Transport Code," *Proceedings of Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2010 (SNA + MC2010)*, Tokyo, Japan, October 17-21, 2010, on CD-ROM (2010).

[4] R. Procassini, D. Cullen, G. Greenman, C. Hagmann, K. Kramer, S. McKinley, M. O'Brien, J. Taylor, "New Capabilities in Mercury: A Modern Monte Carlo Particle Transport Code," *Proceedings of Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007)*, Monterey, California, April 15-17, 2007, on CD-ROM (2007).

[5] R. Procassini, J. Taylor, S. McKinley, G. Greenman, D. Cullen, M. O'Brien, B. Beck, C. Hagmann, "Update on the Development and Validation of Mercury: A Modern, Monte Carlo Particle Transport Code," *Proceedings of Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications*, Avignon, France, September 12-15, 2005, on CD-ROM (2005).

[6] R. J. Procassini, D. E. Cullen, G. M. Greenman, C. A. Hagmann, "Verification and Validation of Mercury: A Modern, Monte Carlo Particle Transport Code," *Proceedings of The Monte Carlo Method: Versatility Unbounded in a Dynamic Computing World*, Chattanooga, Tennessee, April 17-21, 2005, on CD-ROM (2005).

[7] R. Procassini, J. Taylor, I. Corey, J. Rogers, "Design, Implementation and Testing of Mercury: A Parallel Monte Carlo Transport Code," *Proceedings of Nuclear Mathematical and Computational Sciences: A Century in Review, A Century Anew*, Gatlinburg, Tennessee, April 6-11, 2003, on CD-ROM (2003).

[8] J. M. Hall, V. A. Semenov, F. Albert C. P. J. Barty, "Numerical simulation of nuclear materials detection, imaging and assay with MEGa-rays," *Proceedings of INMM, 52nd Annual Meeting*, Palm Desert, California, July 17-21, 2011, on CD-ROM (2011).

[9] R. Buck, E. Lent, T. Wilcox, S. Hadjimarkos, "COG: A Multi-particle Monte Carlo Transport Code," Lawrence Livermore National Laboratory Report UCRL-TM-202590 (2002).

[10] A. B. McKinney, G. W. McKinney, D. B. Pelowitz, B. J. Quiter, A. Coalter, "MCNPX NRF Library - Release 4," *Trans. Am. Nucl. Soc.*, **104**, Hollywood, Florida, June 26-30, 2011, on CD-ROM (2011).

[11] B. Beck, P. Brantley, E. Brooks, F. Daffin, C. Hagmann, S. Quaglioni, J. Rathkopf, "MCAPM-C Generator and Collision Routine (Gen2000/Bang2000) Documentation," Lawrence Livermore National Laboratory Report (2012).

[12] G. M. Greenman, R. J. Procassini, C. J. Clouse, "A Monte Carlo Method for Calculating Initiation Probability," *Proceedings of Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007)*, Monterey, California, April 15-17, 2007, on CD-ROM (2007).

[13] M. E. Rising, F. B. Brown, A. K. Prinja, "The Probability of Initiation in MCNP," *Trans. Am. Nucl. Soc.*, **102**, San Diego, California, June 13-17, 2010, on CD-ROM (2010).

[14] T. E. Booth, "Comments on Monte Carlo Probability of Initiation Estimates for Neutron Fission Chains," *Nucl. Sci. Engr.*, **166**, pp. 175-178 (2010).

[15] T. E. Booth, "Monte Carlo Probability of Initiation Estimates in MCNP," Los Alamos National Laboratory Report LA-UR 09-05874 (2009).

[16] M. S. McKinley, P. S. Brantley, "Probability of Initiation and Extinction in the Mercury Monte Carlo Code," *Proceedings of International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)*, Sun Valley, Idaho, May 5-9, 2013, on CD-ROM (2013).

[17] G. Greenman, M. O'Brien, R. Procassini, K. Joy, "Enhancements to the Combinatorial Geometry Particle Tracker in the Mercury Monte Carlo Transport Code: Embedded Meshes and Domain Decomposition," *Proceedings of International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)*, Saratoga Springs, New York, May 3-7, 2009, on CD-ROM (2009).

[18] R. J. Procassini, M. J. O'Brien, J. M. Taylor, "Load Balancing of Parallel Monte Carlo Transport Applications," *Proceedings of Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications*, Avignon, France, September 12-15, 2005, on CD-ROM (2005).

[19] M. J. O'Brien, P. S. Brantley, K. I. Joy, "Scalable Load Balancing for Massively Parallel Distributed Monte Carlo Particle Transport," *Proceedings of International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)*, Sun Valley, Idaho, May 5-9, 2013, on CD-ROM (2013).

[20] T. A. Brunner, P. S. Brantley, "An Efficient, Robust, Domain-Decomposition Algorithm for Particle Monte Carlo," *J. of Comp. Phys.*, **228**, pp. 3882-3890 (2009).

[21] *International Handbook of Evaluated Criticality Safety Benchmark Experiments*, Nuclear Energy Agency, on CD-ROM (2010).

[22] D. B. Pelowitz, Editor, "MCNP6™ User's Manual," Los Alamos National Laboratory Report LA-CP-11-01708 (2011).

[23] F. R. Metzger, "Resonance Fluorescence in Nuclei," *Prog. in Nuc. Phys.*, **7**, pp. 53-88 (1959).

APPENDIX: NRF CROSS SECTION GENERATION

For a low (less than 4 MeV) energy photon, the interaction of the photon with a nucleus typically involves only nuclear resonance fluorescence (NRF) in which the photon is absorbed by the nucleus and later one or more photons are emitted by the nucleus. The NRF cross section is composed of very narrow ($\ll 1$ eV) resonances. For Monte Carlo transport, the NRF process is divided into absorption, requiring only the cross section, and emission, requiring the branching ratio of the excited nuclear levels.

For a nucleus at rest, absorption can occur when the energy of the photon, $E_{\rm photon}$, and one of the nuclear levels, designated as l, of energy E_l satisfy

$$E_{\rm r} = E_{\rm photon} = E_l \left(1 + \frac{E_l}{2mc^2}\right) \quad , \tag{A-1}$$

where m is the mass of the nucleus in its ground state, c is the speed of light, and $E_{\rm r}$ indicates the resonance energy. For a moving target (see Eq. (4) in [23]), the energy of a photon appears to be $\gamma E_{\rm photon}(1+\beta)$ where $\beta = v/c$, v is the velocity of the target, and $\gamma = 1/\sqrt{(}1-\beta^2)$. Even for a target at a temperature of 300 K, the motion of the target greatly broadens the effective width of each resonance. From Eq. (11) in [23], the cross section of the thermally-broadened resonance is to a good approximation given by

$$\sigma(E_{\rm r}) = \sigma_0 \exp\left(-(E_{\rm photon} - E_{\rm r})^2/\Delta^2\right) \quad , \tag{A-2}$$

where $\Delta = E_{\rm photon}\sqrt{2T/(mc^2)}$, T is the temperature of the target, and σ_0 is defined to be consistent with Eq. (11) in [23].

For emission, we assume that the target comes to rest before the nucleus decays. The energy of a photon in the lab frame emitted from a level of energy E_l is given by

$$E_{\rm photon} = E_l \left(1 - \frac{E_l}{2(mc^2 + E_l)}\right) \quad . \tag{A-3}$$

For ^{238}U, we only considered the first (e1) and second (e2) nuclear levels. As the e1 level has a relatively long half-life compared to the e2 level and its energy is relatively low, the cross section for the e1 level was set to zero. The e1 level was included, however, since the e2 level has a branch to it. A pointwise cross section for the e2 resonance was calculated using Eq. (A-2) to an accuracy of 0.1% over the energy range $E_{\mathrm{r}} - 5\Delta$ to $E_{\mathrm{r}} + 5\Delta$ (requiring 581 cross section values per resonance).

Table VI. ^{238}U NRF Cross Section Data

Level	Energy	E_r	σ_0	Δ	Half-Life	Decay Level	Branching Ratio
	[MeV]	[MeV]	[cm^2]	[MeV]	[s]		
e1	0.044916	N/A	N/A	N/A	2.06×10^{-10}	e0	1.0
e2	0.680110	0.6801110	2.463×10^{-22}	3.284×10^{-7}	3.50×10^{-14}	e0 e1	0.558659 0.441341